

RAE9-99-0073

PATENT



- 1 -

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	:	Before the Examiner:
Alexander, Jr. et al.	:	Philpott, Justin M.
Serial No.: 09/513,518	:	Group Art Unit: 2665
Filed: February 25, 2000	:	
	:	IBM Corporation
Title: PORTABLE NETWORKING	:	Dept. 9CCA/Bddg. 002-2
INTERFACE METHOD AND	:	3039 Cornwallis Road
APPARATUS FOR DISTRIBUTED	:	Research Triangle Park, NC 27709
SWITCHING SYSTEM	:	

APPEAL BRIEF

RECEIVED

MAY 24 2004

Technology Center 2600

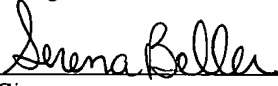
Mail Stop Appeal Brief-Patents  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

I. REAL PARTY IN INTEREST

The real party in interest is International Business Machines Corporation, which is the assignee of the entire right, title and interest in the above-identified patent application.

CERTIFICATION UNDER 37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450, on May 17, 2004.

  
Signature

05/21/2004 AWONDAF1 00000127 500563 09513518

01 FC:1402 330.00 DA

Serena Beller  
(Printed name of person certifying)

II. RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellants, Appellants' legal representative or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 35-68 are pending in the Application. Claims 35-68 stand rejected.

IV. STATUS OF AMENDMENTS

The Appellants' response to the Office Action having a mailing date of August 25, 2003, has been considered, but the Examiner indicated that it did not place the application in condition for allowance because the Appellants' arguments were deemed unpersuasive.

V. SUMMARY OF INVENTION

The proliferation of personal computers, digital telephones, telephony and telecommunications technology has resulted in the development of complex switches in order to efficiently communicate digital data between a number of different devices. Specification, page 2, lines 3-6. These communication systems are generally referred to as networks. Specification, page 2, lines 6-7. Each network operates on the basis of one or more switches which route digital data from an originating device to a destination device. Specification, page 2, lines 7-8. To this end, communication protocols have been developed in order to standardize and streamline communications between devices and promote connectivity. Specification, page 2, lines 8-10.

As advances are made in telecommunications and connectivity technology, additional protocols are rapidly being developed in order to improve the efficiency

and interconnectivity of networking systems. Specification, page 2, lines 11-13. As these advances occur, modifications are required to the switches in order to allow the switches to appropriately deal with the new protocols and take advantage of the new efficiencies that they offer. Specification, page 2, lines 13-16.

Unfortunately, a switch can represent a large capital investment in a network system. Specification, page 2, lines 17-18. The frequency in which new protocols are developed makes it impractical to upgrade switches with every protocol introduced to the market. Specification, page 2, lines 18-19. Accordingly, what is needed is a system and device for improving interface portability within the switch so that switches can be quickly and easily upgraded and new network interface protocols can be written and supported on multiple switch fabrics. Specification, page 2, lines 19-22.

The problems outlined above may at least in part be solved in some embodiments by defining two primary interfaces within the switch. Specification, page 3, lines 3-4. The first interface is called the Forwarding Database Distribution Library (FDDL) Application Program Interface (API). Specification, page 3, lines 4-5. One of the purposes of this interface is to allow each protocol application to distribute its database and functionality to intelligent port controllers within the switch. Specification, page 3, lines 5-7. Such distribution facilitates hardware forwarding at the controller. Specification, page 3, lines 7-8. Each protocol application may define a specific set of FDDL messages that are exchanged between the protocol application and the switch fabric, which passes the messages to software running at each port controller. Specification, page 3, lines 8-11.

The second interface defined by the invention is called the Switch Services API. Specification, page 3, lines 12-13. This interface may be a generic way for controlling data message flow between the ports interfaces and the switch device driver. Specification, page 3, lines 13-14. A set of specific messages may be defined

to allow uniform exchange of information about the hardware status of the port as well as an interface for sending and receiving data frames. Specification, page 3, lines 14-16.

#### VI. ISSUES

Are claims 35-68 properly rejected under 35 U.S.C. §103(a) as being unpatentable over Annaamalai et al. (U.S. Patent No. 6,445,715) (hereinafter "Annaamalai") in view of Hartmann et al. (U.S. Patent No. 6,516,355) (hereinafter "Hartmann")?

#### VII. GROUPING OF CLAIMS

Claims 35, 36, 43, 44, 51, 52, 56, 57, 61 and 62 form a first group.

Claims 37, 41, 45, 49, 53, 58, 63 and 67 form a second group.

Claims 38, 46, 54, 59 and 64 form a third group.

Claims 39, 42, 47, 50, 55, 60, 65 and 68 form a fourth group.

Claims 40, 48 and 66 form a fifth group.

The reasons for these groupings are set forth in Appellants' arguments in Section VIII.

#### VIII. ARGUMENT

- A. The Examiner has not presented any objective evidence for combining Annaamalai with Hartmann.

A *prima facie* showing of obviousness requires the Examiner to establish, *inter alia*, that the prior art references teach or suggest, either alone or in combination, all of the limitations of the claimed invention, and the Examiner must provide a motivation or suggestion to combine or modify the prior art reference to

make the claimed inventions. M.P.E.P. §2142. The showings must be clear and particular. *In re Lee*, 277 F. 3d 1338, 1343, 61 U.S.P.Q.2d 1430, 1433-34 (Fed. Cir. 2002); *In re Kotzab*, 217 F. 3d 1365, 1370, 55 U.S.P.Q.2d 1313, 1317 (Fed. Cir. 2000); *In re Dembiczak*, 50 U.S.P.Q.2d. 1614, 1617 (Fed. Cir. 1999). Broad conclusory statements regarding the teaching of multiple references, standing alone, are not evidence. *Id.*

The Examiner's motivation for modifying Annaamalai with Hartmann to include an FDDL system that comprises a base FDDL system and a plurality of software application towers, as recited in claims 35 and similarly in claims 43, 51, 56 and 61, is "in order to control a number of switches having different message protocols thus providing improved network adaptability." Paper No. 7, pages 7 and 9. This motivation is insufficient to support a *prima facie* case of obviousness as discussed below.

Annaamalai teaches a dynamic trunk protocol that enables dynamic negotiation of trunk encapsulation types between ports connecting intermediate stations in a computer network. Abstract.

Hartmann, on the other hand, teaches a single API that can be used to control a number of switches having different message protocols. Column 3, lines 35-39.

As stated above, the Examiner's motivation for combining Annaamalai with Hartmann is "in order to control a number of switches having different message protocols thus providing improved network adaptability." Paper No. 7, pages 7 and 9. The Examiner's motivation appears to have been gleaned from the secondary reference, Hartmann. In fact, the Examiner cites column 3, lines 27-33 and 36-39 of Hartmann as support for his motivation. Paper No. 7, page 7. This is not evidence as to why one of ordinary skill in the art with the primary reference, Annaamalai, in front of him would have been motivated to modify Annaamalai with the teachings of

the secondary reference, Hartmann. The Examiner's motivation is motivation for the secondary reference, Hartmann, to solve its problem. This is not a suggestion to combine the primary reference, Annaamalai, with the secondary reference, Hartmann. There is no suggestion in Annaamalai of controlling a number of switches having different message protocols (Examiner's motivation). Neither is there a suggestion in Annaamalai of controlling a number of switches having different message protocols thus providing improved network adaptability (Examiner's motivation). The Examiner must provide evidence as to why one of ordinary skill in the art with Annaamalai in front of him, which teaches a dynamic trunk protocol that enables dynamic negotiation of trunk encapsulation types between ports connecting intermediate stations in a computer network, would be motivated to modify Annaamalai with the teachings of Hartmann, which teaches a single API that can be used to control a number of switches having different message protocols. *See In re Lee*, 61 U.S.P.Q.2d 1430, 1433-1434 (Fed. Cir. 2002); *In re Kotzab*, 55 U.S.P.Q.2d 1313, 1318 (Fed. Cir. 2000). Merely stating what the secondary reference teaches is not evidence for combining a primary reference, Annaamalai, with the secondary reference, Hartmann. *See Id.* Consequently, the Examiner's motivation is insufficient to support a *prima facie* case of obviousness for rejecting claims 35-68. *In re Lee*, 61 U.S.P.Q.2d 1430, 1434 (Fed. Cir. 2002).

Further, the Examiner must submit objective evidence and not rely on his own subjective opinion in support of combining Annaamalai, which teaches a dynamic trunk protocol that enables dynamic negotiation of trunk encapsulation types between ports connecting intermediate stations in a computer network, with Hartmann, which teaches a single API that can be used to control a number of switches having different message protocols. *Id.* There is no suggestion in Annaamalai of controlling a number of switches having different message protocols. Neither is there any suggestion in Annaamalai of having a single API that can be used to control a number of switches having different message protocols. Since the Examiner has not

submitted objective evidence for modifying Annaamalai with Hartmann, the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 35-68. *Id.*

Further, the Examiner must submit objective evidence and not rely on his own subjective opinion in support of modifying Annaamalai to include an FDDL system that comprises a base FDDL system and a plurality of software application towers (Examiner admits that Annaamalai does not disclose this limitation). *Id.* There is no suggestion in Annaamalai to have a base FDDL system. Neither is there any suggestion in Annaamalai to have a plurality of software application towers. Since the Examiner has not submitted objective evidence for modifying Annaamalai to include an FDDL system that comprises a base FDDL system and a plurality of software application towers, the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 35-68. *Id.*

As a result of the foregoing, Appellants respectfully assert that the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 35-68. M.P.E.P. §2143.

B. Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest the following limitations.

Appellants respectfully assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "a switch fabric system having circuitry operable to attach to the CPU" as recited in claim 35 and similarly in claims 43 and 61. Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "communicating the information from the port controller to a switch fabric" as recited in claim 51 and similarly in claim 56. The Examiner cites parsing engine 303 of Annaamalai as teaching a switch fabric. Paper No. 7, page 7. Appellants respectfully traverse and assert that Annaamalai instead teaches parsing engine 303 that receives the results from the result bus and drives

aggregate information to a switching bus 310. Column 6, lines 23-26. Annaamalai further teaches that parsing engine 303 may further extract pertinent information from the frames/packets traversing the switching bus. Column 6, lines 26-29. This is not the same as a switch fabric. The Examiner must provide a basis in fact and/or technical reasoning to support the assertion that parsing engine 303 is a switch fabric system. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). The Examiner must provide extrinsic evidence that must make clear that parsing engine 303 is a switch fabric system. *In re Robertson*, 169 F.3d 743, 745 (Fed. Cir. 1999). Since the Examiner has not supported the assertion that parsing engine 303 is a switch fabric system, the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 35, 43, 51, 56 and 61. M.P.E.P. §2143.

Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "a switch device driver operable to execute on the CPU" as recited in claim 35 and similarly in claim 61. Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "a switch driver means for communicating with the FDDL means and the port controller" as recited in claim 43. Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "communicating the information from the switch fabric to a switch device driver within an operating system" as recited in claim 51 and similarly in claim 56. The Examiner cites forwarding database 330 of Annaamalai as teaching a switch device driver. Paper No. 7, page 8. Appellants respectfully traverse and assert that Annaamalai instead teaches a layer 2 forwarding engine 330 configured to access and process information stored in forwarding database 332. Column 6, lines 16-19. This is not the same as the switch device driver. The Examiner must provide a basis in fact and/or technical reasoning to support the assertion that layer 2 forwarding engine 330 is a switch device driver. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). The Examiner must provide extrinsic evidence that must make clear that



layer 2 forwarding engine 330 is a switch device driver. *In re Robertson*, 169 F.3d 743, 745 (Fed. Cir. 1999). Since the Examiner has not supported the assertion that layer 2 forwarding engine 330 is a switch device driver, the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 35, 43, 51, 56 and 61. M.P.E.P. §2143.

Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "wherein the software application is operable to communicate with the FDDL system, the FDDL system is operable to communicate with the switch device driver, and the switch device driver is operable to communicate with the switch fabric" as recited in claim 35 and similarly in claims 43 and 61. Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "communicating the information from the switch device driver to a Forwarding Database Distribution Library (FDDL)" as recited in claim 51 and similarly in claim 56. For at least the reasons stated above, Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest a switch fabric and a switch device driver. Hence, Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest an FDDL system operable to communicate with a switch device driver. Further, Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest a switch device driver operable to communicate with the switch fabric. Further, Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest communicating the information from a switch device driver to a FDDL. Therefore, the Examiner has not presented a *prima facie* case of obviousness, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "wherein the FDDL system defines an FDDL

API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver" as recited in claim 37 and similarly in claims 41, 45, 49, 53, 58, 63 and 67. The Examiner cites column 7, lines 13-67 of Annaamalai as teaching the above-cited claim limitation. Paper No. 7, page 9. Appellants respectfully traverse and assert that Annaamalai instead teaches a DTP protocol that enables dynamic negotiation of trunk encapsulation types between local and neighbor ports interconnecting switches in a computer network. This language is not the same as an FDDL system that defines an FDDL API for communication for a software application. Further, this language is not the same as an FDDL system that defines a Switch Services API for communication with a switch device driver. In fact, the passage cited by the Examiner does not even mention forwarding database 332 which the Examiner had previously indicated as teaching an FDDL system. Therefore, the Examiner has not presented a *prima facie* case of obviousness, since the Examiner is relying upon an incorrect factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

The Examiner, in response to the above argument, states:

[W]hile Annaamalai may not recite specifically 'FDDL API' and 'Switch Services API', as discussed in the previous office action Annaamalai teaches utilizing a plurality of application program interfaces (APIs) (e.g., col. 7, lines 13-67 regarding DTP protocol wherein messages are exchanged between application and the switching fabric). Furthermore, these APIs communicate with the FDDL system via forwarding engine 330 coupled to forwarding database 332 (e.g., see col. 7, lines 31-44 and Fig. 3). Thus, Annaamalai teaches the FDDL system (e.g., forwarding database 332) defines APIs (e.g., via operations by forwarding engine 330) for communication with the software application (e.g., see col. 5, lines 49-64 and col. 7, lines 31-44 regarding software) and for communication with the switch device driver (e.g., forwarding engine 330). Paper No. 7, page 3.

Appellants respectfully traverse the assertion that Annaamalai teaches an FDDL system that defines APIs. There is no language in the cited passage of Annaamalai that discloses the term FDDL or API. Further, as stated above, there is no language in the cited passage of Annaamalai that discloses forwarding database 332 (Examiner asserts that forwarding database 332 teaches an FDDL system) or discloses forwarding database 332 defining APIs via forwarding engine 330. The Examiner is simply relying upon his own subjective opinion which is insufficient to support a *prima facie* case of obviousness. M.P.E.P. §2143. The Examiner must provide a basis in fact and/or technical reasoning to support the assertion that the cited passage teaches defining APIs by an FDDL system. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). The Examiner must provide extrinsic evidence that the cited passage teaches defining APIs by an FDDL system. *In re Robertson*, 169 F.3d 743, 745 (Fed. Cir. 1999). Since the Examiner has not supported the assertion that the cited passage teaches defining APIs by an FDDL system, the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 37, 41, 45, 49, 53, 58, 63 and 67. M.P.E.P. §2143.

Further, there is no language in the cited passage of Annaamalai that discloses defining an FDDL API for communication with the software application or defining an API for communication with the switch device driver. Further, as stated above, there is no language in the cited passage of Annaamalai that teaches forwarding database 332 or teaches forwarding database 332 defining APIs for communication with a software application or a switch device driver. There is no language in the cited passage of Annaamalai that teaches communicating with a software application or with forwarding engine 330 (Examiner asserts that forwarding engine 330 teaches a switch device driver). The Examiner must provide a basis in fact and/or technical reasoning to support the assertion that the cited passage teaches defining an FDDL API for communication with the software application and defining an API for communication with the switch device driver. *Ex parte Levy*, 17 U.S.P.Q.2d 1461,

1464 (Bd. Pat. App. & Inter. 1990). The Examiner must provide extrinsic evidence that the cited passage teaches defining an FDDL API for communication with the software application and defining an API for communication with the switch device driver. *In re Robertson*, 169 F.3d 743, 745 (Fed. Cir. 1999). Since the Examiner has not supported the assertion that the cited passage teaches defining an FDDL API for communication with the software application and defining an API for communication with the switch device driver, the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 37, 41, 45, 49, 53, 58, 63 and 67. M.P.E.P. §2143.

Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "wherein the FDDL system defines an FDDL API for communication with the software application and the second software application, and the FDDL system defines a Switch Services API for communication with the switch device driver" as recited in claim 38 and similarly in claims 46, 54, 59 and 64. The Examiner cites column 7, lines 13-67 and column 5, lines 53-60 of Annaamalai as teaching the above-cited claim limitation. Paper No. 7, page 9. For at least the reasons stated above, Appellants respectfully assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest an FDDL system that defines an FDDL API for communication with a first and a second software application. For at least the reasons stated above, Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest an FDDL system that defines a switch services API for communication with the switch device driver. Therefore, the Examiner has not presented a *prima facie* case of obviousness, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "wherein the FDDL system comprises: a base FDDL system; a software application tower FDDL system; and a second software

application tower FDDL system" as recited in claim 39 and similarly in claims 42, 47, 50, 55, 60, 65 and 68. The Examiner cites logical management device 120 of Hartmann as teaching a base FDDL system and element 100 of Hartmann as teaching an FDDL system. Paper No. 7, page 8. Appellants respectfully traverse and assert that Hartmann instead teaches that element 100 corresponds to a switching engine with the ability to read from and write to persistent configuration files which are particular to a given switch and which contains configuration and maintenance functions. Column 5, lines 1-10. A switching engine is not the same as an FDDL which defines a set of APIs designed to enable protocol forwarding functions to be distributed in the manner that is simple, efficient and deportable. Specification, page 7, lines 1-3. Further, logical device management 120 of Hartmann provides the logic necessary to manage per-device information. Column 5, lines 33-34. This is not the same as a base FDDL system which may be used to translate a command. Therefore, the Examiner has not presented a *prima facie* case of obviousness, since the Examiner is relying upon on an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "wherein the base FDDL communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system" as recited in claim 39 and similarly in claims 42, 47, 50, 55, 60, 65 and 68. The Examiner cites a call controlled transaction manager 116 of Hartmann as teaching a switch device driver. Paper No. 7, page 9. The Examiner further cites logical device management 120 of Hartmann as teaching a base FDDL system that communicates with an object server interface translator 124 representing a software application and a media API translator 112 as representing a

second software application tower FDDL system. Paper No. 7, page 8. Further, the Examiner cites an object server interface translator 124 of Hartmann as teaching both a software application and a software application tower FDDL system. As recited in the above-cited claim limitation, a software application and a software application tower FDDL system are two separate elements. Hence, Hartmann does not teach a software application communicating with a software application tower FDDL system. Similarly, the Examiner cites a media IPA translator 112 of Hartmann as teaching both a second software application and a second software application tower FDDL system. A second software and a second software application tower FDDL system are two separate elements. Thus, Hartmann does not teach a second software application that communicates with a second software application tower FDDL system. Further, as stated above, logical device management 120 of Hartmann is not the same as a base FDDL system. Hence, Hartmann does not teach a base FDDL system that communicates with a software application tower FDDL system and a second software application tower FDDL system. Therefore, the Examiner has not presented a *prima facie* case of obviousness, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

The Examiner states, in response to the above argument, that an object server coupled to element 124 corresponds to a software application tower FDDL system and that an OA&M interface translation at element 124 corresponds to a software application. Paper No. 7, page 4. Appellants respectfully assert that the Examiner is transmogrifying the meaning of Hartmann in order to deduce that Hartmann teaches the above-cited claim limitation. Hartmann teaches that an object server interface translator 124 provides the logic necessary to convert object server API messages from the MMI into native switch "operation, administration, and maintenance" (OA&M) messages and to convert native OA&M messages into object server API messages. Column 5, lines 46-50. There is no language in Hartmann to suggest that

object server interface translator 124 is coupled to an object server where that object server is a software application tower FDDL system. Further, there is no language in Hartmann to suggest that object server interface translator 124 is coupled to an OA&M interface translation where that OA&M interface translation is a software application. The Examiner must provide a basis in fact and/or technical reasoning to support the assertion Hartman teaches an object server coupled to element 124 that corresponds to a software application tower FDDL system and that Hartmann teaches an OA&M interface translation at element 124 that corresponds to a software application. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). The Examiner must provide extrinsic evidence that Hartmann teaches an object server coupled to element 124 that corresponds to a software application tower FDDL system and that Hartmann teaches an OA&M interface translation at element 124 that corresponds to a software application. *In re Robertson*, 169 F.3d 743, 745 (Fed. Cir. 1999). Since the Examiner has not supported the assertion that Hartmann teaches an object server coupled to element 124 that corresponds to a software application tower FDDL system and that Hartmann teaches an OA&M interface translation at element 124 that corresponds to a software application, the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 39, 42, 47, 50, 55, 60, 65 and 68. M.P.E.P. §2143.

Similarly, the Examiner states, in response to the above argument, that a media API coupled to element 112 corresponds to a software application tower FDDL system and that native switch translation at element 112 corresponds to a software application. Paper No. 7, page 5. Appellants respectfully assert that the Examiner is transmogrifying the meaning of Hartmann in order to deduce that Hartmann teaches the above-cited claim limitation. There is no language in Hartmann to suggest that media API translator 112 is coupled to a media API where that media API is a software application tower FDDL system. Further, there is no language in Hartmann to suggest that media API translator 112 is coupled to a native switch translation

where that native switch translation is a software application. The Examiner must provide a basis in fact and/or technical reasoning to support the assertion that Hartmann teaches a media API coupled to element 112 that corresponds to a software application tower FDDL system and that Hartmann teaches a native switch translation at element 124 in Hartmann that corresponds to a software application. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). The Examiner must provide extrinsic evidence that Hartmann teaches a media API coupled to element 112 that corresponds to a software application tower FDDL system as well as that Hartmann teaches a native switch translation at element 124 that corresponds to a software application. *In re Robertson*, 169 F.3d 743, 745 (Fed. Cir. 1999). Since the Examiner has not supported the assertion that Hartmann teaches a media API coupled to element 112 that corresponds to a software application tower FDDL system as well as that Hartmann teaches a native switch translation at element 124 that corresponds to a software application, the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 39, 42, 47, 50, 55, 60, 65 and 68. M.P.E.P. §2143.

Appellants further assert that Annaamalai and Hartmann, taken singly or in combination, do not teach or suggest "an independent software application shim operable to execute on the CPU, wherein the independent software application communicates with the independent software application shim and the independent software application shim communicates with the switch device driver" as recited in claim 40 and similarly in claims 48 and 66. The Examiner cites native switch translation 110, 112, 114, 124 of Hartmann as teaching a shim application. Paper No. 7, page 9. Appellants respectfully traverse and assert that Hartmann instead teaches a connection API translator 110, a media API translator 112, and an in-band signaling API translator 114 for communicating with the call control entity 104. Column 5, lines 15-19. Hartmann further teaches an object server interface translator 124 which provides the logic necessary to convert object server API messages from the man-machine interface into native switch "operation, administration and maintenance



messages" and to convert native "operation, administration and maintenance messages" into object server API messages. Column 5, lines 45-50. These are not shim applications. A shim application may refer to a software component that interfaces between two other software components. Therefore, the Examiner has not presented a *prima facie* case of obviousness, since the Examiner is relying upon an incorrect, factual predicate in support of the rejection. *In re Rouffet*, 47 U.S.P.Q.2d 1453, 1455 (Fed. Cir. 1998).

Further, the Examiner must provide a basis in fact and/or technical reasoning to support the assertion that elements 110, 112, 114, 124 of Hartmann correspond to shim applications. *Ex parte Levy*, 17 U.S.P.Q.2d 1461, 1464 (Bd. Pat. App. & Inter. 1990). The Examiner must provide extrinsic evidence that elements 110, 112, 114, 124 of Hartmann correspond to shim applications. *In re Robertson*, 169 F.3d 743, 745 (Fed. Cir. 1999). Since the Examiner has not supported the assertion that elements 110, 112, 114, 124 of Hartmann correspond to shim applications, the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 40, 48 and 66. M.P.E.P. §2143.40.

As a result of the foregoing, Appellants respectfully assert that there are numerous claim limitations not taught or suggested in the cited prior art, and thus the Examiner has not presented a *prima facie* case of obviousness for rejecting claims 35-68 as being unpatentable over Annaamalai in view of Hartmann. M.P.E.P. §2143.

IX. CONCLUSION

For the reasons noted above, the rejections of claims 35-68 are in error. Appellants respectfully request reversal of the rejections and allowance of claims 35-68.

Respectfully submitted,

WINSTEAD SECHREST & MINICK P.C.

Attorneys for Appellants

By: 

Robert A. Voigt, Jr.  
Reg. No. 47,159  
Kelly K. Kordzik  
Reg. No. 36,571

P.O. Box 50784  
Dallas, Texas 75201  
(512) 370-2832

**APPENDIX**

35. A network switch comprising:
- a CPU;
  - a memory system having circuitry operable to attach to the CPU;
  - a switch fabric system having circuitry operable to attach to the CPU;
  - a port controller having circuitry operable to attach to the switch fabric system;
  - a software application operable to execute on the CPU;
  - a Forwarding Database Distribution Library (FDDL) system operable to execute on the CPU; and
  - a switch device driver operable to execute on the CPU,
- wherein the software application is operable to communicate with the FDDL system, the FDDL system is operable to communicate with the switch device driver, and the switch device driver is operable to communicate with the switch fabric.
36. The network switch of claim 35 further comprising a second software application operable to execute on the CPU, wherein the second software application communicates with the FDDL system.
37. The network switch of claim 35 wherein the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.
38. The network switch of claim 36 wherein the FDDL system defines an FDDL API for communication with the software application and the second software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

39. The network switch of claim 36 wherein the FDDL system comprises:  
a base FDDL system;  
a software application tower FDDL system; and  
a second software application tower FDDL system  
wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.
40. The network switch of claim 35 further comprising:  
an independent software application operable to execute on the CPU; and  
an independent software application shim operable to execute on the CPU,  
wherein the independent software application communicates with the independent software application shim and the independent software application shim communicates with the switch device driver.
41. The network switch of claim 40 further comprising a second software application operable to execute on the CPU, wherein the FDDL system defines an FDDL API for communication with the software application and the second software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.
42. The network switch of claim 40 wherein the FDDL system comprises:  
a base FDDL system;  
a software application tower FDDL system; and  
a second software application tower FDDL system  
wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software

application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.

43. A network switch comprising:
- a CPU;
  - a memory system having circuitry operable to attach to the CPU;
  - a switch fabric system having circuitry operable to attach to the CPU;
  - a port controller having circuitry operable to attach to the switch fabric system;
  - a protocol means for providing a service to a network system;
  - a Forwarding Database Distribution Library (FDDL) means for communicating with the protocol means; and
  - a switch device driver means for communicating with the FDDL means and the port controller.
44. The network switch of claim 43 further comprising a second protocol means for providing a second service to the network system, wherein the FDDL means communicates with the second protocol means.
45. The network switch of claim 43 wherein the FDDL means defines an FDDL API for communication with the software application, and the FDDL means defines a Switch Services API for communication with the switch device driver.
46. The network switch of claim 44 wherein the FDDL means defines an FDDL API for communication with the protocol means and the second protocol means, and the FDDL system defines a Switch Services API for communication with the switch device driver means.
47. The network switch of claim 44 wherein the FDDL means comprises:
- a base FDDL means for communicating with the switch device driver means;

a protocol tower FDDL means for communicating with the protocol means and the base FDDL means; and

a second protocol tower FDDL means for communicating with a second protocol means and the base FDDL means.

48. The network switch of claim 43 further comprising:

an independent protocol means for providing an independent service to the network system; and

an independent protocol shim for communicating with the independent protocol means and the switch device driver means.

49. The network switch of claim 48 further comprising a second protocol means for providing a second service to the network system, wherein the FDDL means communicates with the second protocol means.

50. The network switch of claim 48 wherein the FDDL means comprises:

a base FDDL means for communicating with the switch device driver means;

a protocol tower FDDL means for communicating with the protocol means and the base FDDL means; and

a second protocol tower FDDL means for communicating with the second protocol means and the base FDDL means.

51. A method of providing communications over a network system utilizing a first protocol and a second protocol, the method comprising the steps of:

receiving information at a port controller in a first protocol from a first node machine;

communicating the information from the port controller to a switch fabric;

communicating the information from the switch fabric to a switch device driver within an operating system;

communicating the information from the switch device driver to a Forwarding Database Distribution Library (FDDL); and

communicating the information from the FDDL to a first protocol client.

52. The method of claim 51 further comprising the steps of:

receiving additional information at a port controller in a second protocol from a first node machine;

communicating the additional information from the port controller to a switch fabric;

communicating the additional information from the switch fabric to a switch device driver within an operating system;

communicating the additional information from the switch device driver to a Forwarding Database Distribution Library (FDDL); and

communicating the additional information from the FDDL to a second protocol client.

53. The method of claim 52 wherein

all communicating between the switch device driver to the FDDL is done through a switch services API; and

all communicating from the FDDL to the first protocol client and the second protocol client is done through an FDDL API.

54. The method of claim 52 further comprising the steps of:

defining a switch services API for communication between the switch device driver; and

defining an FDDL API for communication between the first protocol client and the FDDL.

55. The method of claim 52 further comprising the steps:

receiving the information from the switch device driver at an FDDL base within the FDDL;

passing the information from the FDDL base to a first protocol FDDL tower within the FDDL; and

sending the information from the first protocol FDDL tower to the first protocol client.

56. A computer-readable medium having stored thereon computer-executable instructions for performing the steps comprising:

receiving information at a port controller in a first protocol from a first node machine;

communicating the information from the port controller to a switch fabric;

communicating the information from the switch fabric to a switch device driver within an operating system;

communicating the information from the switch device driver to a Forwarding Database Distribution Library (FDDL); and

communicating the information from the FDDL to a first protocol client.

57. The computer-readable medium of claim 56 having further stored thereon computer-executable instructions for performing the steps comprising:

receiving additional information at a port controller in a second protocol from a first node machine;

communicating the additional information from the port controller to a switch fabric;

communicating the additional information from the switch fabric to a switch device driver within an operating system;

communicating the additional information from the switch device driver to a Forwarding Database Distribution Library (FDDL); and

communicating the additional information from the FDDL to a second protocol client.



58. The computer-readable medium of claim 57 wherein  
all communicating between the switch device driver to the FDDL is done through a switch services API; and  
all communicating from the FDDL to the first protocol client and the second protocol client is done through an FDDL API.
59. The computer-readable medium of claim 57 having further stored thereon computer-executable instructions for performing the steps comprising:  
defining a switch services API for communication between the switch device driver; and  
defining an FDDL API for communication between the first protocol client and the FDDL.
60. The computer-readable medium of claim 57 having further stored thereon computer-executable instructions for performing the steps comprising:  
receiving the information from the switch device driver at an FDDL base within the FDDL;  
passing the information from the FDDL base to a first protocol FDDL tower within the FDDL; and  
sending the information from the first protocol FDDL tower to the first protocol client.
61. A network system comprising:  
a network switch comprising a CPU, a memory system having circuitry operable to attach to the CPU, a switch fabric system having circuitry operable to attach to the CPU a port controller having circuitry operable to attach to the switch fabric system, a software application operable to execute on the CPU, a Forwarding Database Distribution Library (FDDL) system operable to execute on the CPU, and a switch device driver operable to execute on the CPU, wherein the software application is operable to communicate with the FDDL system, the FDDL system is

operable to communicate with the switch device driver, and the switch device driver is operable to communicate with the switch fabric;

a backbone; and

a workstation,

wherein the workstation is logically connected to the backbone, and

wherein the backbone is logically connected to the port controller of the network switch.

62. The network system of claim 61 further comprising a second software application operable to execute on the CPU, wherein the second software application communicates with the FDDL system.

63. The network system of claim 61 wherein the FDDL system defines an FDDL API for communication with the software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

64. The network system of claim 62 wherein the FDDL system defines an FDDL API for communication with the software application and the second software application, and the FDDL system defines a Switch Services API for communication with the switch device driver.

65. The network system of claim 62 wherein the FDDL system comprises:  
a base FDDL system;  
a software application tower FDDL system; and  
a second software application tower FDDL system  
wherein the base FDDL system communicates with the switch device driver, the software application communicates with the software application tower FDDL system, the second software application communicates with the second software application tower FDDL system, and the base FDDL system communicates with the software application tower FDDL system and the second software application tower FDDL system.

66. The network system of claim 61 further comprising:  
an independent software application operable to execute on the CPU; and  
an independent software application shim operable to execute on the CPU,  
wherein the independent software application communicates with the  
independent software application shim and the independent software application shim  
communicates with the switch device driver.

67. The network system of claim 66 further comprising a second software  
application operable to execute on the CPU, wherein the FDDL system defines an  
FDDL API for communication with the software application and the second software  
application, and the FDDL system defines a Switch Services API for communication  
with the switch device driver.

68. The network system of claim 66 wherein the FDDL system comprises:  
a base FDDL system;  
a software application tower FDDL system; and  
a second software application tower FDDL system  
wherein the base FDDL system communicates with the switch device driver,  
the software application communicates with the software application tower FDDL  
system, the second software application communicates with the second software  
application tower FDDL system, and the base FDDL system communicates with the  
software application tower FDDL system and the second software application tower  
FDDL system.